2. Basic of Algorithms Analysis

"For me, great algorithms are the poetry of computation. Just like verse, they can be terse, allusive, dense, and even mysterious. But once unlocked, they cast a brilliant new light on some aspect of computing." - Francis Sullivan

Computational Tractability

Worst case running time. Obtain bound on largest possible running time of algorithm on input of a given size $N$, and see how this scales with $N$.

- Generally captures efficiency in practice.
- Draconian view, but hard to find effective alternative.

Desirable scaling property. When the input size increases by a factor of 2, the algorithm should only slow down by some constant factor $C$.

There exists constants $c > 0$ and $d > 0$ such that on every input of size $N$, its running time is bounded by $cN^d$ steps.

Def. An algorithm is efficient if it has polynomial running time.

Justification. It really works in practice!
Asymptotic Order of Growth

Upper bounds. \( T(n) \) is \( O(f(n)) \) if there exist constants \( c > 0 \) and \( n_0 \geq 0 \) such that for all \( n \geq n_0 \) we have \( T(n) \leq c \cdot f(n) \).

Lower bounds. \( T(n) \) is \( \Omega(f(n)) \) if there exist constants \( c > 0 \) and \( n_0 \geq 0 \) such that for all \( n \geq n_0 \) we have \( T(n) \geq c \cdot f(n) \).

Tight bounds. \( T(n) \) is \( \Theta(f(n)) \) if \( T(n) \) is both \( O(f(n)) \) and \( \Omega(f(n)) \).

Example: \( T(n) = 32n^2 + 17n + 32 \).
- \( T(n) \) is \( O(n^2) \), \( O(n^3) \), \( \Omega(n^2) \), \( \Omega(n) \), and \( \Theta(n^2) \).
- \( T(n) \) is not \( O(n) \), \( \Omega(n^3) \), \( \Theta(n) \), or \( \Theta(n^3) \).

Asymptotic Bounds for Some Common Functions

Polynomials. \( a_0 + a_1 n + \ldots + a_d n^d \) is \( \Theta(n^d) \) if \( a_d > 0 \).

Polynomial time. Running time is \( O(n^d) \) for some constant \( d \) independent of the input size \( n \).

Logarithms. \( O(\log_a n) = O(\log_b n) \) for any constants \( a, b > 0 \).

Logarithms. For every \( x > 0 \), \( \log n = O(m^x) \).

Exponentials. For every \( r > 1 \) and every \( d > 0 \), \( n^d = O(r^n) \).
Linear Time: $O(n)$

**Linear time.** Running time is at most a constant factor times the size of the input.

**Computing the maximum.** Compute maximum of $n$ numbers $a_1, \ldots, a_n$.

```plaintext
max \leftarrow a_1
for i = 2 to n {
    if (a_i > max)
        max \leftarrow a_i
}
```

**Linear arithmetic time.** Arises in divide-and-conquer algorithms.

**Sorting.** Mergesort and heapsort are sorting algorithms that perform $O(n \log n)$ comparisons.

**Largest empty interval.** Given $n$ time-stamps $x_1, \ldots, x_n$ on which copies of a file arrive at a server, what is largest interval of time when no copies of the file arrive?

**$O(n \log n)$ solution.** Sort the time-stamps. Scan the sorted list in order, identifying the maximum gap between successive time-stamps.

---

Quadratic Time: $O(n^2)$

**Quadratic time.** Enumerate all pairs of elements.

**Closest pair of points.** Given a list of $n$ points in the plane $(x_1, y_1), \ldots, (x_n, y_n)$, find the pair that is closest.

**$O(n^2)$ solution.** Try all pairs of points.

```plaintext
min \leftarrow (x_1 - x_2)^2 + (y_1 - y_2)^2
for i = 1 to n {
    for j = i+1 to n {
        d \leftarrow (x_i - x_j)^2 + (y_i - y_j)^2
        if (d < min)
            min \leftarrow d
    }
}
```

**Remark.** $\Omega(n^2)$ seems inevitable, but this is just an illusion. — Chapter 5

---

Cubic Time: $O(n^3)$

**Cubic time.** Enumerate all triples of elements.

**Set disjointness.** Given $n$ sets $S_1, \ldots, S_n$ each of which is a subset of $1, 2, \ldots, n$, is there some pair of these which are disjoint?

**$O(n^3)$ solution.** For each pairs of sets, determine if they are disjoint.

```plaintext
foreach set S_i {
    foreach other set S_j {
        foreach element p of S_i {
            if (no element of S_i belongs to S_j)
                report that S_i and S_j are disjoint
        }
    }
}
```
Polynomial Time: $O(n^k)$ Time

**Independent set of size $k$.** Given a graph, are there $k$ nodes such that no two are joined by an edge?

$O(n^k)$ solution. Enumerate all subsets of $k$ nodes.

```plaintext
foreach subset $S$ of $k$ nodes {
  check whether $S$ is an independent set
  if ($S$ is an independent set)
    report $S$ is an independent set
}
```

- Check whether $S$ is an independent set $= O(k^2)$.
- Number of $k$ element subsets $= \binom{n}{k} = \frac{n!}{(n-k)!k!} = \frac{n(n-1)(n-2)\cdots(n-k+1)}{k(k-1)(k-2)\cdots(2)(1)} \leq \frac{n^k}{k!}$ assuming $k$ is a constant.

Exponential Time

**Independent set.** Given a graph, what is maximum size of an independent set?

$O(n^2 2^n)$ solution. Enumerate all subsets.

```plaintext
S* ← ∅
foreach subset $S$ of nodes {
  check whether $S$ is an independent set
  if ($S$ is largest independent set seen so far)
    update $S* ← S$
}
```